VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Uživatelská webová rozhraní pomocí technologie triády PHP, Apache, MySQL

User web interface using triad PHP, Apache, MySQL

2013 Kateřina Vopálková

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Zadání bakalářské práce

Student: **Kateřina Vopálková**

Studijní program: B2649 Elektrotechnika

Studijní obor: 3901R039 Biomedicínský technik

Téma: Uživatelská webová rozhraní pomocí technologie triády PHP, Apache, MySQL
User Web Interface using Triad PHP, Apache, MySQL

Zásady pro vypracování:

1. Seznámení se s programovacím jazykem PHP.
2. Seznámení se s principem fungování HTTP serveru Apache.
3. Seznámení se s databázovým serverem MySQL a možností práce s ním pomocí programovacího jazyku PHP.
4. Seznámení se s nástrojem pro vytváření internetových aplikací Nette Framework.
5. Vytvoření internetové aplikace pomocí triády PHP, Apache, MySQL a za pomocí Nette Framework, která vytvoří administrativní rozhraní nad MySQL databází, která bude obsahovat data dle konceptu IOT (Internet Of Things) z nemocničního prostředí urgentního příjmu Fakultní nemocnice v Ostravě-Porubě. Tato aplikace pak bude obsahovat uživatelské prvky pro správu entit obsažených v této databázi (editace, mazání, jednoduché a složené vyhledávací dotazy).
6. Testování aplikace na rychlost odezvy a počet operací, které bude schopná provést v čase v závislosti na komplexnosti dotazu a počtu procházených záznamů. Testování také rychlosti prováděných asynchronních operací AJAX a rychlosti zpracování dat pomocí javascriptu.
7. Optimalizace aplikace a použitých ovladačů pro přístup k databázi dle výsledků testů . Přístup k databázi a použitý ovladač je volitelný. Je však doporučena ORM (Objektově relační mapování) Dibi a ovladač PHP MySQL. Pro práci s Javascriptem a AJAX je doporučena knihovna jQuery.
8. Zhodnocení dosažených výsledků.

Seznam doporučené odborné literatury:

[1] BORONCZYK, Tim. *PHP 6, MySQL, Apache : vytváříme webové aplikace.* Brno: Computer Press, 2009. ISBN 978-80-251-2767-4.
[2] PROCHÁZKA, David. *PHP 6 : začínáme programovat.* Praha: Grada Publishing, 2012. ISBN 978-80-247-3899-4.
[3] POŠMURA, Vlastimil. *Apache : příručka správce WWW serveru.* Praha: Computer Press, 2002. ISBN 80-7226-696-9.
[4] DUBOIS, Paul. *MySQL profesionálně : komplexní průvodce použitím, programováním a správou MySQL.* Vyd. 1. Překlad Jan POKORNÝ. Brno: Mobil Media, 2003. 1071 s. ISBN 80-86593-41-X.
[5] PÍSEK, Slavoj. *JavaScript : efektní nástroj oživení www stránek.* Praha: Grada Publishing, 2001. ISBN 80-247-0014-X.
[6] DARIE, Cristian et al. *AJAX a PHP : tvoříme interaktivní webové aplikace profesionálně.* Vyd. 1. Překlad Roman SKŘIVÁNEK. Brno: Zoner Press, 2006. 320 s. ISBN 80-86815-47-1.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jakub Jirka**

Datum zadání: 16.11.2012

Datum odevzdání: 07.05.2013

doc. Ing. Jiří Koziorek, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Abstrakt

Tato práce se zaměřuje na možnosti vytváření webových aplikací a vytvoření konkrétní aplikace pracující nad databází nemocničního informačního systému. V první části práce obsahuje teoretický popis programovacího jazyka PHP, serveru Apache, databázového systému MySQL a frameworku Nette. Dále práce obsahuje stručný popis použitého softwaru – vývojového prostředí Eclipse a webových serverů EasyPHP. V další části práce se nachází teoretický návrh následován popisem reálných částí vytvořené aplikace.

Klíčová slova

PHP, Apache, MySQL, Nette Framework, webová aplikace, databáze, administrativní rozhraní, uživatelské webové rozhraní

Abstract

This thesis focuses on possibilities of creating web applications and on creating a concrete application, which is able to work with a database from information system in hospital. The first part contains the theoretical description of PHP programming language, Apache server, MySQL database system and Nette Framework. The thesis also contains a brief description of software, which was used to create this application – software development environment Eclipse and web servers EasyPHP. Next part of this thesis is about theoretical design of the application followed by description of real parts of created application.

Key words

PHP, Apache, MySQL, Nette Framework, web application, database, administrative interface, user web interface

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně. Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

V Ostravě dne 7. května 2013

. . . . . .

Kateřina Vopálková

Poděkování

Děkuji vedoucímu mé bakalářské práce Ing. Jakubu Jirkovi za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

Seznam použitých zkratek

- **AJAX:** Asynchronous JavaScript and XML, technologie pro vytváření interaktivních webových aplikací
- **AMP:** triáda Apache, MySQL, PHP
- **API:** Application Programming Interface, rozhraní pro programování aplikací
- **HTTP:** HyperText Transfer Protocol, protokol pro přenos HTML stránek
- **HTTPS:** HTTP Secure, zabezpečené HTTP
- **HTML:** HyperText Markup Language, jazyk webového prohlížeče
- **MySQL:** databázový systém používající dotazovací jazyk SQL
- **MVC:** model-view-controller, koncept rozdělení aplikace na tři samostatné vrstvy
- **MVP:** model-view-presenter, koncept MVC v Nette Framework
- **PHP:** původně Personal Home Page, dnes PHP: Hypertext Preprocessor, programovací jazyk webových aplikací
- **RFID:** Radio Frequency Identification, čipy pro čtení a zápis pro bezkontaktní komunikaci
- **SSL:** Secure Socket Layer, vrstva bezpečných socketů
- **UML:** Unified Modeling Language, jazyk pro vytváření diagramů

Obsah

1 Úvod

Cílem této bakalářské práce je vytvoření webové aplikace pomocí prostředků Apache, PHP a MySQL. Tato aplikace má vytvářet rozhraní nad databází nemocničního informačního systém na urgentním příjmu Fakultní nemocnice Ostrava. Teoretickým úkolem je seznámit se programovacím jazykem PHP, jeho aplikací ve frameworku Nette, s konceptem rozdělení aplikace do tří samostatných vrstev model, view a presenter. Dále se seznámit s databázovým jazykem MySQL, naučit se zapisovat dotazy nad databází také pomocí jazyka PHP a seznámit se s funkcí serveru Apache.

Praktická část této bakalářské práce má obsahovat analytický návrh aplikace, jenž by nejvíce vyhovoval webové aplikaci, která by se reálně využívala na pracovišti urgentního příjmu. Dále v závislosti na této analýze takovou aplikaci naprogramovat za pomoci znalostí získaných studiem výše zmíněných programátorských prostředků v Nette Framework. Tato aplikace by měla být schopná spojit se s databází a získávat potřebná data, tato data následně zobrazit pomocí tabulky a měla by umožnit dále se záznamy pracovat. Mezi vyžadované funkce aplikace patří přidávání, úprava, mazání, seřazování a vyhledávání záznamů. V neposlední řadě je třeba celou aplikaci otestovat a optimalizovat, aby i při velkém počtu záznamů v databázi byla schopná rychle a efektivně pracovat a zůstala tak uživatelsky přívětivou.

2 Webové aplikace

Pod pojmem webové aplikace rozumíme aplikace, které využívají internet, nebo intranet. Aplikace jsou spouštěny pomocí webového klienta (webový prohlížeč), jenž komunikuje se serverem a je schopný přijatá data ze serveru zobrazovat. Protikladem webových aplikací jsou tzv. desktopové aplikace.

Mezi největší výhody webových aplikací oproti desktopovým patří především jejich snadná dostupnost, jednoduchost, flexibilita a finanční nenáročnost. Tyto aplikace fungují na všech operačních systémech a jsou přístupné všem uživatelům přes jakýkoliv webový prohlížeč.

Základem přenosu dat v rámci internetu (intranetu) je protokol HTTP (HyperText Transfer Protocol), jehož první verze byla vyvinuta v roce 1991. Protokol HTTP je nejčastěji používaným protokolem, pracuje s hypertextovými dokumenty formátu HTML (HyperText Markup Language). Webový server zasílá klientovi soubor HTML značek, které klient dokáže zpracovat a následně zobrazit. Při používání pouze HTTP a HTML máme ovšem značně omezené možnosti realizace webových stránek. Proto postupem času začaly přibývat nové technologie jak na straně klienta, tak na straně serveru.

Technologie na straně klienta, mezi které patří dnes nejpoužívanější JavaScript, Java aplety a Macromedia Flash, jsou vhodné především pro implementaci grafických prvků (flash animace), nebo v případě JavaScriptu ke kontrole formulářových dat bez nutnosti odeslání na server.

Nejvýznamnějšími technologiemi na straně serveru jsou PHP (viz kapitola PHP), MySQL (viz kapitola MySQL) a jejich konkurenční produkty. Díky těmto technologiím je server schopný kromě samotného odeslání HTML souboru provádět výpočty, pracovat s databázemi, používat objektově orientované programování, implementační logiku atd.

[1]

Jednou z nejoblíbenějších serverových technologií je platforma AMP (zkratka pro Apache, MySQL, PHP). Jedná se o open source technologii, neboli o technologii s otevřeným zdrojovým kódem. Open source v praxi pro uživatele znamená, že software je zdarma a je flexibilní pro použití na jakékoliv platformě. Z programátorského hlediska je každý oprávněn ke změnám zdrojového kódu (odtud název open source) a úpravám softwaru dle vlastních potřeb. AMP je tedy volně dostupná varianta serveru, která je tvořena samotným webovým serverem Apache (viz kapitola Apache), interpretem jazyka PHP a databázovým serverem MySQL.

[2]

Systém fungování platformy AMP lze dobře demonstrovat na příkladu fungování restaurace z knihy Timothyho Boronczyka *PHP 6, MySQL, Apache: Vytváříme webové aplikace*: „Jakmile host (návštěvník webové aplikace) přijde do vaší restaurace (webová aplikace), posadí

se a objedná si jídlo se specifickými požadavky (vyžádá si konkrétní stránku či zdroj) – například aby byl steak propečený tak akorát (medium well). Číšník (Apache) zanese tyto specifické požadavky do kuchyně a předá je šéfkuchaři (PHP). Ten se poté odebere do skladu (MySQL), odkud vezme suroviny (data), z nichž připraví pokrm, který nakonec předá na talíři (webová stránka) zpět číšníkovi. Ten jej následně naservíruje zákazníkovi přesně podle jeho objednávky.“ [2]

Každou ze tří složek AMP lze nainstalovat nezávisle na zbylých dvou, ale především se využívají jako celek. Chceme-li použít všechny složky zároveň, je přesto nutné každou zvlášť nainstalovat, konfigurovat, případně testovat. To může být pro nezkušeného uživatele i přes existenci spousty knih a návodu problematické, proto lze tuto situaci vyřešit instalací softwaru, pod kterým lze spustit celou platformu AMP. Mezi tyto programy patří např. EasyPHP, WampServer a jiné. Uživateli se tak nabízí jednoduché rozhraní pro správu celého serveru pro vývoj webové aplikace.

2.1 PHP

PHP (neboli PHP: Hypertextový preprocesor) je skriptovací jazyk určený k programování dynamických webových aplikací. Lze jej však využít i pro programování desktopových aplikací. PHP patří mezi technologie na straně serveru, kde se provádějí skripty a na stranu klienta je přenesena výsledná stránka zapsaná v HTML (popř. s JavaScriptem). Zdrojový kód stránky obsahuje základní tagy HTML a kód PHP, který je uzavřen mezi značky `<?php` a `?>`.

Výhody PHP:

- nejpoužívanější skriptovací jazyk na internetu
- jazyk specializovaný na tvorbu webových aplikací
- umožňuje práci s databázemi a soubory
- funguje na libovolném operačním systému
- jednoduchá syntaxe

2.1.1 Historie PHP

První verze PHP, napsaná v roce 1994 Rasmusem Lerdorfem, byla původně určena k mapování návštěv na Lerdorfově stránce. Tento soubor skriptů byl pojmenován „Personal Home Page Tools“. Postupem času bylo vyžadováno více prvků, a tak Lerdorf vytvořil nový model, který umožňoval práci s databází a poskytoval framework pro vytváření jednoduchých dynamických aplikací. V roce 1995 se PHP stal oficiálně open source projektem. Po významném předělání v dubnu roku 1996 se PHP prakticky stalo samostatným skriptovacím jazykem. Později téhož roku byla tato verze označena jako PHP 2.0. Mezi lety 1997–1998 se počet domén, na nichž bylo použito PHP, vyšplhal z několika málo na 60 tisíc, což v té době bylo zhruba 1 % všech domén.

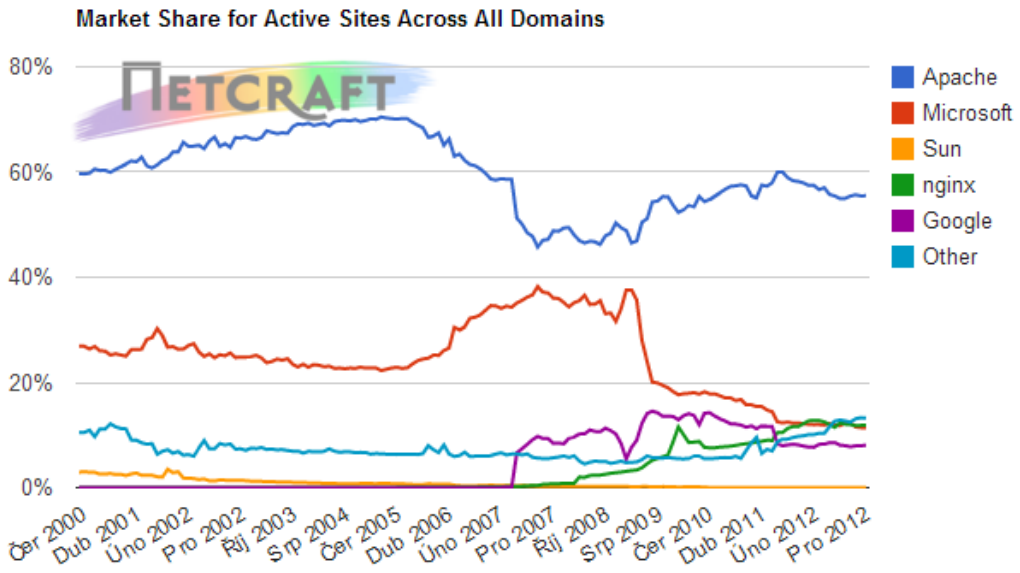
Jádro PHP v roce 1999 přepracovali Andi Gutmans a Zeev Suraski. Vznikla tak verze PHP 4, jejímž jádrem se stal Zend Engine (název vznikl spojením jmen jeho autorů). Nejnovější verzí je PHP 5 (v současnosti nejaktuálnější PHP 5.4.10 a beta verze PHP 5.5), která byla zveřejněna v roce 2004.

[3]

2.2 Apache

Apache (neboli projekt pod názvem *Apache HTTP Server Project*) je v současné době nejpoužívanějším webovým serverem. V lednu 2013 tento server používalo 55 % všech webových aplikací, tj. cca 103 mil. Graf vývoje podílu různých serverů na trhu lze vidět na Obr. 2.1. Projekt jako takový zahrnuje vývoj spolehlivého kvalitního a mnoha funkcemi vybaveného open source kódu, který implementuje webový server. Server Apache je zdarma a je použitelný pro jakýkoliv operační systém.

[4], [5], [6]



Obr. 2.1: Vývoj podílu serverů na trhu aktivních stránek na všech doménách. [4]

2.2.1 Architektura Apache

Architektura Apache se skládá z několika vrstev. Primární vrstvu tvoří samotný operační systém (Linux, Windows, MacOS…). Nad operačním systémem se nachází vrstva, která zajišťuje komunikaci mezi operačním systémem a síťovým prostředím. Tato vrstva je tvořena hlavním programem Apache. Součástí hlavního programu je jádro, vestavěné moduly a několik standardních knihoven. Jedním z vestavěných modulů je modul *http_core*, který spolu s jádrem využívá základní funkčnost Apache. Třetí vrstvu Apache tvoří moduly, které jsou propojeny s API (Application Programming Interface, česky rozhraní pro programování aplikací) pomocí

jádra. Samotné moduly jsou rozšířením funkcí Apache, které nejsou nezbytné, ale rozšiřují základní Apache o různé funkce. V poslední vrstvě architektury serveru Apache se mohou nacházet další moduly, které dále rozšiřují možnosti Apache. Zde patří například moduly, které mohou pracovat s moduly jiných výrobců (např. *mod_perl*).

Mezi moduly Apache verze 2.2 patří například tyto:

- základní a multiprocessingové moduly (*core*...)
- moduly pro vytvoření prostředí
- moduly pro mapování URL
- moduly pro řízení přístupu
- moduly pro tvorbu dynamického obsahu
- moduly pro vývoj

[6]

2.2.2 Apache a bezpečnost

V dnešní době se celkem často setkáváme s útoky na webové servery. Některé z nich mají za úkol změnit obsah dokumentů, některé z nich získat citlivé údaje, nebo dokonce poškodit operační systém. Standardem je přijímání a odesíláni dat servery bez zabezpečení. Zdrojové kódy stránek lze volně zobrazit a uživatelský přístup je anonymní. Proto existuje snaha webové servery chránit před útoky různými způsoby. Zabezpečení serveru můžeme zajistit těmito metodami:

- šifrování
- autorizovaný přístup, zabezpečení přístupu
- oddělení serveru od internetu

Existuje několik možností šifrování. Lze šifrovat přenosové linky, nebo dokumenty. Mezi nejznámější a nejvíce využívané šifrovací protokoly patří HTTPS (HTTP Secure) a SSL (Secure Socket Layer).

Při použití autorizovaného přístupu je na serveru uložen soubor s přihlašovacími údaji uživatelů (jméno a heslo). Pokud chce uživatel zobrazit chráněný dokument, je pomocí formuláře vyzván k zadání přihlašovacích údajů. Autorizovaný přístup je možno nastavit přes konfiguraci serveru Apache nebo i jinými způsoby (např. pomocí vlastního přihlašovacího formuláře).

[6]

2.3 MySQL

MySQL je celosvětově nejpoužívanější open source databázový software využívající dotazovací jazyk SQL, resp. jeho dialekt. Největší výhoda MySQL spočívá v jeho rychlosti, dostupnosti a snadném používání. Obecně databázový server slouží k uchování a následné práci s daty. Data jsou v databázi uložena ve formě tabulek, v nichž řádky odpovídají jednotlivým záznamům a sloupce obsahují část záznamu stejného typu (jméno, datum atd.).

Mezi základní datové typy sloupců patří:

- char – jakýkoliv znak
- varchar – posloupnost znaků (řetězec)
- int – celočíselné hodnoty
- int unsigned – celočíselné kladné hodnoty
- text – lze vložit text do 65 536 znaků
- decimal – desetinní čísla
- enum – může nabývat 2 hodnot (např. *true* nebo *false*)
- date – datum ve formátu YYYY-MM-DD
- time – čas ve formátu hh:mm:ss
- datetime – datum a čas ve formátu YY-MM-DD hh:mm:ss
- a další

[2]

Jednotlivé buňky v tabulce mohou mít také hodnotu NULL, což znamená, že jsou prázdné – zadaný parametr není vyplněn. Je vhodné, když každá tabulka v databázi obsahuje speciální sloupec ID (identifikační číslo). V rámci tabulky má každý záznam unikátní ID, které zároveň slouží jako primární klíč.

Pro připojení k databázi MySQL potřebujeme znát název hostitele a uživatelské jméno a heslo. Při použití lokálního serveru je název hostitele *localhost*.

V jazyce PHP existuje pro práci s MySQL řada funkcí. Zde je několik z nich:

- *mysql_connect* se připojuje k serveru MySQL
- *mysql_query($dotaz[, $zdroj])* odesílá příkaz na server, v proměnné *dotaz* může být uloženo např. „*SELECT * FROM tabulka*“
- *mysql_fetch_rows* vrací řádek dat ve formě pole

Kromě základních funkcí lze vytvářet i funkce složitější. Takové mohou být složité vyhledávací dotazy, filtrování dat aj.

2.4 Nette

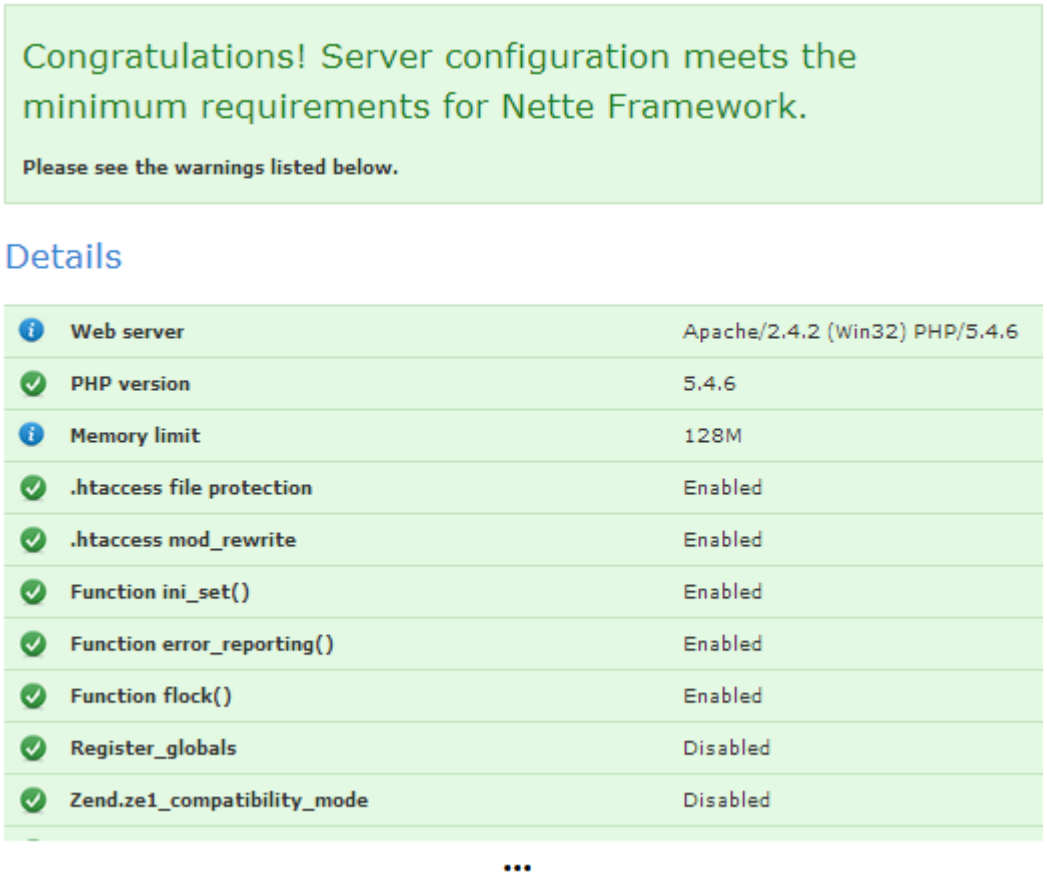
S příchodem PHP 5 se ve velké míře začínají vytvářet tzv. frameworky. Framework je softwarová konstrukce napsaná v určitém jazyce (zde v PHP), která programátorovi usnadňuje práci tím, že obsahuje knihovny a poskytuje tak spoustu užitečných funkcí. Díky tomu také eliminuje rozsah kódu, který musí programátor napsat pro vytvoření webové aplikace. Jedním z nejznámějších PHP frameworků je *Zend Framework*, který má licenci open source.

V České republice je nejpoužívanějším frameworkem *Nette Framework*, jehož autorem je David Grudl. V současné době je tento framework používán k vytváření a správě webových aplikací také velkých firem v České republice. Mezi největší výhody frameworku Nette patří:

- podpora moderních technologií
- rozsáhlá uživatelská dokumentace

- klade důraz na zabezpečení
- velká podpora českých vývojářů
- vysoká rychlost
- obsahuje ladicí nástroje (tzv. laděnku)
- díky použití konceptu MVC je přehledný
- dobrá práce s formuláři
- automaticky ošetřuje validaci pomocí JavaScriptu

K používání stačí uživateli stáhnout knihovnu Nette, kterou je třeba umístit do adresáře s webovou aplikací, a ověřit si systémové požadavky. Především je třeba mít nainstalovaný webový server. Pro ověření systémových požadavků slouží soubor *checker.php* uložený v adresáři *http://localhost/Nette/tools/Requirements-checker*, který vše zkontroluje, popř. uživatele upozorní výjimky nebo chyby. Výsledek testu systémových požadavků je na Obr. 2.2.



Obr. 2.2: Ukázka výsledku testu systémových požadavků Nette Framework.

2.4.1 Model-view-controller

Většina z dnes používaných PHP frameworků jsou založený na konceptu MVC neboli model-view-controler. Principem tohoto konceptu je oddělení kódu aplikační logiky (model) od kódu, který zobrazuje data, (view) a od kódu obsluhy (controller). Použití MVC aplikaci jednak zpřehledňuje a pak také umožňuje testovat a ladit každou část zvlášť. Model obsahuje aplikační logiku (byznys logika) a data. View zobrazuje model, výsledky získané s modelu (model navenek poskytuje pevně dané rozhraní). Controller pracuje s požadavky uživatele. Uži-

vatel tedy zašle požadavek, který je controllerem zpracován, zavolá se model, kde se např. něco spočítá, nebo vyhledá, a nakonec se zavolá view, kterým se zobrazí konečný výsledek. Nette Framework používá presentery, což je obdoba controllerů. V konceptu MVP (model-view-presenter) je prezentační logika uložena v presenteru. Lze říci, že presenter má zde roli prostředníka mezi modelem a view.

[7]

2.4.2 Adresářová struktura

Při vytváření webových aplikací existuje doporučená adresářová struktura. Její podoba je následující:

- app/
  - model/
  - presenters/
  - templates/
  - config.neon
  - bootstrap.php
- libs/
  - Nette/
  - autoload.php
- log/
- temp/
- www/
  - index.php

Složka *app/* obsahuje samotnou aplikaci, jejíž součástí je rozdělení na model, view (složka *templates/*) a presenter, hlavní konfigurační soubor *config.neon* a zaváděcí soubor aplikace *bootstrap.php*. Ve složce *libs/*, která je adresářem pro knihovny, najdeme framework Nette a soubor *autoload.php*, kterým se framework načítá a konfiguruje. Následující adresáře jsou pro logy, error logy v případě *log/*, resp. pro dočasné soubory a cache v případě *temp/*. Posledním adresářem je *www/*, který je jako jediný veřejný. V něm je uložen soubor *index.php* umožňující spuštění aplikace, popř. další složky (složky pro obrázky, složky pro kaskádové styly…).

[7]

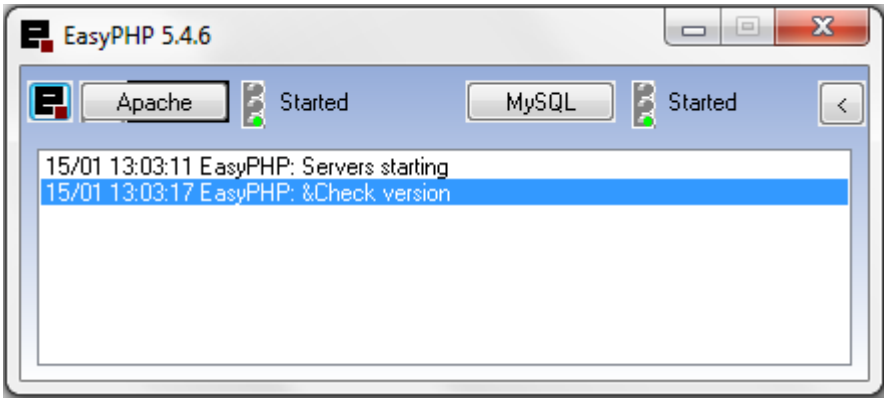
2.5 Použitý software

2.5.1 Eclipse

Eclipse je vývojové prostředí primárně určené pro Java programátory. Díky pluginům lze toto prostředí rozšířit i pro programování v PHP, C++ atd. Existuje několik verzí této platformy – Indigo, Helios, Galileo, Juno a nejnovější verze Eclipse Kepler.

2.5.2 EasyPHP

Program EasyPHP řeší nutnost instalace webového serveru na lokálním počítači. Po instalaci tohoto programu se Apache a MySQL servery spouští tímto programem. Značně tak usnadňuje práci s instalací, konfigurací a spouštěním celého webového serveru. Pokud uživatel do webového prohlížeče napíše *localhost* zobrazí se mu složky a soubory uložené ve složce *www*, která se nachází v adresáři, kde je EasyPHP nainstalováno. Vzhled uživatelského rozhraní EasyPHP je zobrazen na Obr. 2.3.



Obr. 2.3: Uživatelské rozhraní EasyPHP.

3 Návrh aplikace

Na oddělení urgentního příjmu Fakultní nemocnice v Ostravě-Porubě je třeba především dokumentovat pacienty a úkony vykonávané s nimi. V rámci zkvalitnění péče o pacienty, lepší dohledatelnosti záznamů a také ušetření práce nemocničnímu personálu se vytváří snaha papírovou dokumentaci převádět do digitální formy. Produkt této bakalářské práce by měla být aplikace, která umožňuje správu záznamů z nemocničního systému. Tato aplikace má komunikovat s databází, s níž zároveň může komunikovat mobilní zařízení. Databáze primárně pracuje s mobilním zařízením, které informace načítá přes RFID kód. RFID slouží k bezkontaktní komunikaci mezi mobilním zařízením a výrobkem (přístroje, nástroje, pomůcky aj.). Příkladem použití může být přivezení pacienta na urgentní příjem. V tom okamžiku je pomocí pacientova RFID kódu do databáze načteno, kdy byl přivezen.

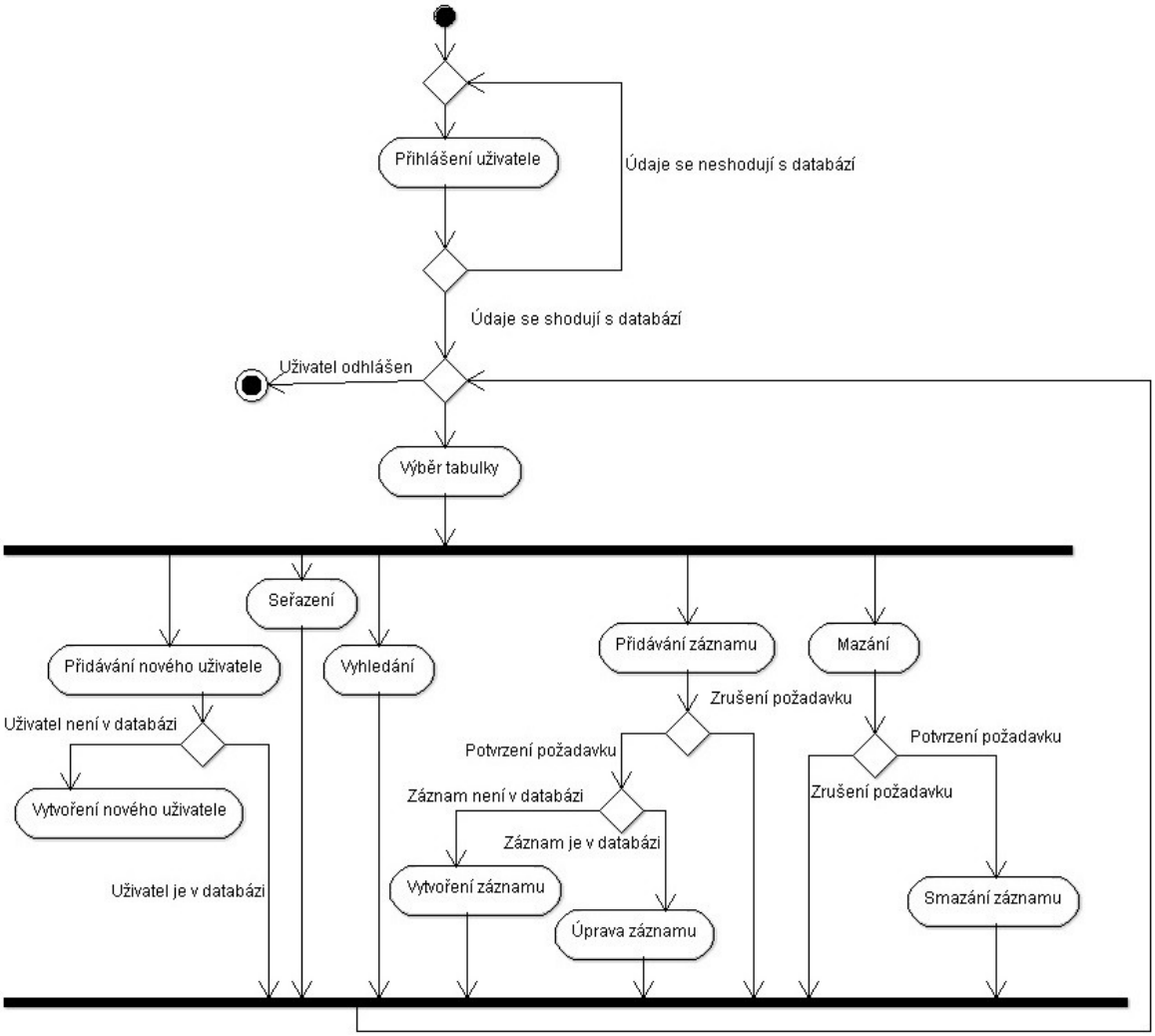
Tato webová aplikace má spravovat různé entity použité v databázi nemocničního systému tak, aby zde bylo možno upravovat nejen pacientskou dokumentaci. Cílem je vytvořit generickou aplikaci. Je totiž pravděpodobné, že postupem času může docházet ke změnám nemocniční databáze. Pokud by se tak stalo a aplikace by nebyla generická, přestala by fungovat. I proto je třeba použít speciálních funkcí, které tuto vlastnost zajistí. Především se jedná o funkci, která automaticky načítá názvy databázových sloupců v tabulkách.

Nemocniční péče je zaměřena především na pacienta, proto i zde byla jako výchozí zvolena tabulka pacientů, v níž jsou viditelné všechny základní údaje o pacientovi (rodné číslo, jméno, příjmení atd.). Z této tabulky by mělo být možné přepnout se do libovolné jiné tabulky (tabulka léků, přístrojů…). Zároveň v tabulce pacientů je nezbytná možnost zobrazit si kartu pacienta, v níž se budou nacházet tabulky s výkony, které budou představovat vazební tabulky mezi jednotlivými základními tabulkami (např. vazební tabulka vyšetření zajišťuje spojení tabulky pacienta s tabulkou přístrojů – reálně se jedná o provedení vyšetření určitého pacienta určitým přístrojem). Co se týká zadaných údajů, je třeba ověřovat platnost rodných čísel a také kontrolovat zadávané záznamy s databází. Nemělo by být tedy možné přidat do databáze dva pacienty se stejným rodným číslem, stejně tak dva zaměstnance se stejným identifikačním číslem.

Celá aplikace by měla být zabezpečená, to znamená, že bez přihlášení uživatele nesmí být přístupné citlivé údaje nejen o pacientech. Jelikož má být aplikace zabezpečená globálně (neboli všechny její části) je nejlepší ošetřit přihlašování hned při spuštění aplikace, kde se zobrazí přihlašovací okno, a bez přihlášení se uživatel dál nedostane. Je také vhodné použít automatické odhlašování při delší neaktivitě uživatele.

Jako zásadní funkce aplikace se považuje její schopnost pracovat s konkrétními záznamy: přidávat, upravovat, mazat, seřazovat a vyhledávat pomocí jednoduchých i komplexních dotazů. Toto by mělo být realizováno prostřednictvím formulářových prvků.

3.1 Diagram aktivit

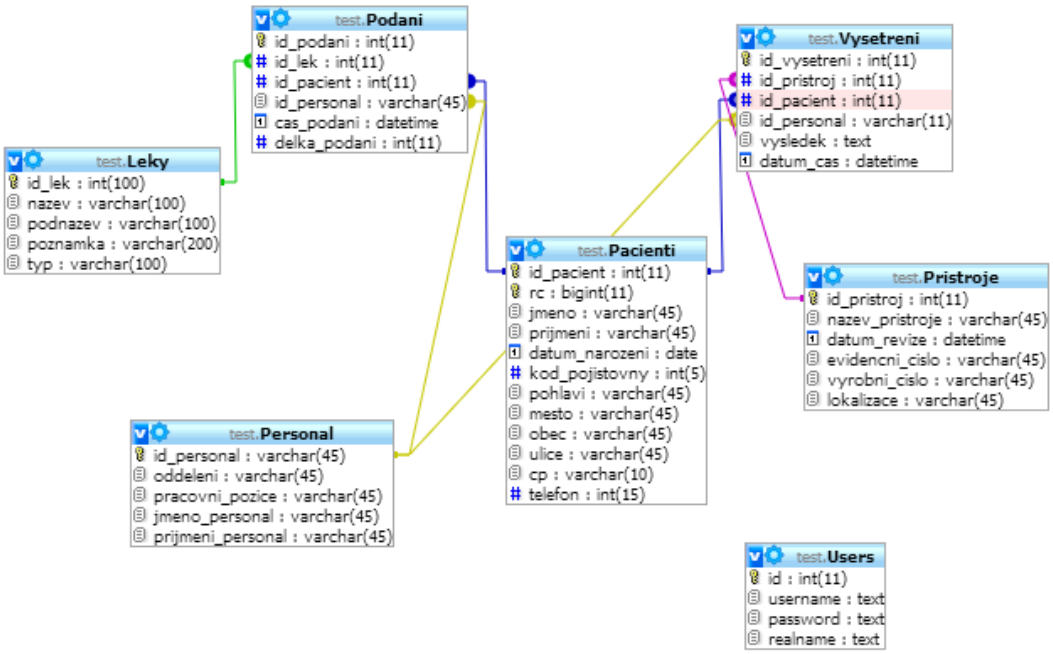


Obr. 3.1: Diagram aktivit navrhované aplikace.

Navržený diagram aktivit dokumentuje aktivity webové aplikace. Pro jeho vytvoření byl použit freeware program ArgoUML, který umožňuje modelaci UML diagramů.

3.2 Testovací databáze

Aplikace bude vytvářena nejdříve nad testovací databází, jejíž struktura je na Obr. 3.2. Tato databáze obsahuje nejzákladnější tabulky a vazby, ze kterých lze vycházet pro vývoj aplikace. Pokud by se měla aplikace nasazovat na složitější databázi, přidaly by se soubory pro zpracování těchto tabulek (modely, presentery i šablony), jejichž kostra by však byla velmi podobná se soubory vytvořenými pro testovací databázi.



Obr. 3.2: Struktura testovací databáze.

Na Obr. 3.2 je možné vidět i tzv. relace neboli vazby mezi jednotlivými tabulkami. Význam tabulky *Users* spočívá v tom, že jsou zde uloženy přihlašovací údaje do aplikace. I proto tato tabulka nemá vazby na jiné tabulky.

4 Realizace aplikace

4.1 Spuštění aplikace

Při spuštění serveru je viditelná pouze složka *www* obsahující soubor *index.php*, kterým se celá aplikace spouští. Aplikace je tedy chráněna proti zobrazení zdrojových kódů cizími uživateli. V souboru *index.php* jsou definovány cesty k samotné aplikaci, ke knihovnám a cesta k veřejnému webovému adresáři. Je zde také volán soubor *bootstrap.php* nacházející se v adresáři *app*. V bootstrapu se zavádí soubor pro načtení knihoven (soubor *autoload.php* v adresáři *libs*), zavádí se konfigurační soubor a určuje se výchozí presenter (zde presenter *Dashboard*). V konfiguračním souboru *config.neon* se deklaruje připojení k databázi, použité modely, aj.

Kód souboru *index.php*, který jako jediný *.php* soubor je zobrazitelný na serveru:

```
<?php

// absolute filesystem path to this web root
define('WWW_DIR', __DIR__);

// absolute filesystem path to the application root
define('APP_DIR', WWW_DIR . '/../app');

// absolute filesystem path to the libraries
define('LIBS_DIR', WWW_DIR . '/../libs');

// load bootstrap file
require APP_DIR . '/bootstrap.php';
```

Ukázka z konfiguračního souboru:

```
common:
      php:
            date.timezone: Europe/Prague

      nette:
            database:
                  dsn: 'mysql:host=rfidexpert.vsb.cz;dbname=test'
                  user: 'vop0003'
                  password: 'rfofthings'

      services:
            authenticator: Authenticator
            pacienti: Pacienti
            users: Users
            leky: Leky
            pristroje: Pristroje
            personal: Personal
            vysetreni: Vysetreni
            podani: Podani
```

V části *database* se v konfiguračním souboru nastavují údaje pro připojení k databázi, v části *services* jsou zahrnuty všechny použité modely.

Načtení knihoven má na starosti *autoload.php*, který si zavolá zaváděcí soubory jednotlivých knihoven Nette a Grido pomocí těchto příkazů:

```
require __DIR__ . '/../libs/Nette/loader.php';
require __DIR__ . '/../libs/Grido/loader.php';
```

Soubory *loader.php* zajistí nahrání potřebných souborů knihovny pro použití v aplikaci. Knihovna Nette obsahuje kompletní Nette Framework, knihovna Grido byla použita pro vytváření tabulek. Po načtení knihoven se spustí aplikace příkazem v bootstrapu

```
$container->application->run();
```

4.2 Koncept MVP v této aplikaci

Jelikož se tato aplikace zakládá na frameworku Nette, splňuje koncept MVP, neboli rozdělení částí aplikace na model, view (šablona) a presenter (viz Model-view-controller).

4.2.1 Presentery

Součástí presenterů je prezentační logika. Pomocí metod jsou zde vytvářeny formuláře a tabulky. Presentery získávají pomocí modelů data, která následně zobrazí pomocí šablon. Základním presenterem této aplikace je *DashboardPresenter.php*, který pracuje se záznamy pacientů a se záznamy výkonů, které byly pacientovi provedeny, a záznamy léků, které byly pacientovi podány (karta pacienta). Kromě funkcí, které vytvářejí a pracují s formuláři a tabulkami, se v presenterech nacházejí funkce, které určují činnosti po načtení konkrétních šablon a těmto šablonám mohou předávat různé parametry (např. id pacienta). Speciální funkcí je funkce *startup()*. Tato funkce určuje, co se stane po načtení daného presenteru. V této aplikaci se zde ověřuje, zda je uživatel přihlášen. V případě vypršení doby přihlášení je uživatel přesměrován na úvodní přihlašovací obrazovku (metoda *setExpiration()*). Funkce *startup()* vypadá takto:

```
protected function startup()
{
  parent::startup();

  if (!$this->user->isLoggedIn()) {
    $this->user->setExpiration('+ 20 minutes', TRUE);
    if ($this->user->logoutReason ===
    Nette\Http\UserStorage::INACTIVITY) {
      $this->flashMessage('Byl jste odhlášen z důvodu neaktivity.
      Přihlaste se, prosím, znovu.');
    }
    if ($this->user->logoutReason ===
    Nette\Http\UserStorage::BROWSER_CLOSED) {
      $this->flashMessage('Byl jste automaticky odhlášen. Přihlaste
      se, prosím, znovu.');
    }
    $this->redirect('Sign:in', array('backlink' => $this-
     >storeRequest()));
  }
}
```

4.2.2 Šablony

Šablony (view) jsou psány latte zápisem. Latte v podstatě sjednocuje zápis PHP a HTML v jednom souboru. Struktura souboru je HTML, stejně tak i některé další prvky stránky (nadpisy aj.) Prvky, které nejsou v jazyce HTML, se uzavírají do složených závorek. V šabloně tak lze používat i např. cyklus foreach. Zápisu ve složených závorkách se říká makro. V šablonách této aplikace jsou makra používána především pro zobrazení formulářů a datagridů.

Základní šablonou celé aplikace je soubor *@layout.latte*. V hlavičce jsou definovány meta parametry, titulek stránky a cesty k souboru se styly *site*.css a ke skriptu, který validuje formuláře v Nette. Tělo stránky je rozděleno na několik úseků. V prvním úseku se zobrazují zprávy uživateli (flash messages), v druhém úseku se zobrazuje obsah stránky. Po přihlášení uživatele se pod obsahem nachází text s informací o přihlášeném uživateli a s možností vytvořit nového uživatele.

V dalších šablonách je zapsáno, co se má nacházet v obsahu dané stránky. Příklad šablony pro přihlašování (*app/templates/Sign/in.latte*):

```
{block #content}

<h1 n:block="title">Přihlášení</h1>

{control signInForm}
```

Tato šablona si ze svého presenteru (*SignPresenter.php*) nechá vykreslit formulář pro přihlašování. Zpracování tohoto formuláře dále probíhá výhradně přes presenter.

4.2.3 Modely

Součástí adresáře s modely je soubor *Authenticator.php* zajišťující přihlašování uživatelů, dále pak základní model *Repository.php*. V tomto základním modelu je definováno spojení s databází, vyhledání všech dat v tabulkách. Funkce *insert* umožňuje vkládání dat do databáze. Tato funkce se volá při odesílání vyplněných formulářů. Dále se zde nachází funkce *getColumnName*, která ze zadané tabulky získá názvy sloupců. Proto při změně v databázi není třeba měnit zdrojový kód pro zobrazení dat.

Ze základního modelu *Repository* dědí ostatní modely reprezentující jednotlivé databázové tabulky. Každá tabulka použitá v databázi má svůj vlastní model, který aplikaci předává její data.

4.3 Přihlašování uživatelů

Jakmile je aplikace spuštěna, zobrazí se přihlašovací formulář (Obr. 4.1). Uživatel je vyzván k zadání svého uživatelského jména a hesla. Po kliknutí na potvrzovací tlačítko se vyplněné údaje ověří pomocí *Authenticator.php*, který se nachází v modelu aplikace.

Obr. 4.1: Přihlašovací obrazovka.

V souboru *Authenticator.php* se nacházejí tři funkce. První je pro načtení databáze. Druhá funkce hledá v databázi v tabulce Users zadané uživatelské jméno a ověřuje heslo (viz ukázka kódu níže). V případě špatného uživatelského jména nebo hesla, je do formuláře vypsána zpráva pro uživatele o neplatnosti jména, nebo hesla. Poslední funkcí je funkce nazvaná *generateHash*. Tato funkce je použita kvůli bezpečnosti hesla. V databázi jsou uložena zašifrovaná hesla. Pokud uživatel zadá své heslo, toto se zašifruje a ověřuje se jeho shodnost se zašifrovaným heslem v databázi. Stejného principu se využívá i při vytváření nových uživatelů.

Ověřovací funkce v souboru *Authenticator.php*:

```
public function authenticate(array $credentials)
{
    list($username, $password) = $credentials;
      $row = $this->database->table('Users')->where('username',
            $username)->fetch();

    if (!$row) {
          throw new Security\AuthenticationException('Neplatné
            uživatelské jméno.', self::IDENTITY_NOT_FOUND);
      }

      if ($row->password !== $this->generateHash($password, $row-
           >password)) {
            throw new Security\AuthenticationException('Neplatné
             heslo.', self::INVALID_CREDENTIAL);
      }

      unset($row->password);
      return new Security\Identity($row->id, NULL, $row->toArray());
}
```

4.4 Tabulky

Základní tabulkou aplikace je tabulka pacientů. Přepínání na další tabulky je realizováno pomocí formuláře, ve kterém si uživatel pomocí prvku select vybere, kterou tabulku chce zobrazit. Po potvrzení se načte šablona pro danou tabulku a zobrazí se obsah příslušné stránky.

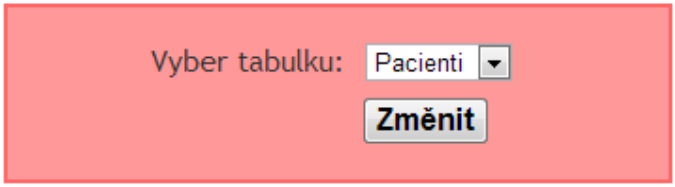
Ukázka z funkce volané po potvrzení formuláře s výběrem tabulky – použití switche:

```
switch (join(" ",$value)) {
      case 'Leky':
            $this->redirect('Leky:');
```

```
        break;
    case 'Pristroje':
        $this->redirect('Pristroje:');
        break;
    case 'Personal':
        $this->redirect('Personal:');
        break;
```

Tento switch se nachází presenteru pro výchozí tabulku pacientů (*DashboardPresenter.php)*. Metoda *redirect* zajišťuje přesměrování.

Výsledný prvek vypadá v aplikaci následovně.



Obr. 4.2: Výběr tabulky.

Všechny tabulky v aplikaci jsou vytvořeny pomocí tzv. datagridu. Zde byl použit datagrid Grido (autor Petr Bugyík). Jedná se o knihovnu navrženou pro Nette Framework, která výrazným způsobem ulehčuje programátorovi práci s tabulkami. Ve funkci, která grid vytváří, stačí napsat jeden příkaz pro vytvoření nového gridu a jeden příkaz pro nastavení modelu, se kterým grid pracuje.

```
$grid = new Grido\Grid($this, $name);
$grid->setModel($this->context->pacienti->findAll());
```

Pro vytvoření nového sloupce zobrazené tabulky je třeba zadat příkaz

```
$grid->addColumn().
```

Metodě *addColumn()* je třeba předat některé povinné parametry, jakými jsou název sloupce, popisek sloupce, typ sloupce (text, datum a další). Pro správnou funkci datagridu je také zapotřebí označit, který sloupec představuje primární klíč databázové tabulky. Toto se provede příkazem:

```
$grid->setPrimaryKey($nazevSloupce);
```

kde proměnná *nazevSloupce* představuje název sloupce v databázové tabulce. Jako parametr metody *setPrimaryKey()* lze zapsat název tohoto sloupce přímo jako řetězec (např. 'id_pacient'), zde je však kvůli zachování genericity aplikace použita proměnná *nazevSloupce*.

Použití datagridu umožňuje jednoduše nastavovat seřazovací a filtrovací dotazy. Pokud chce programátor použít seřazování či filtrování, stačí při vytváření nového sloupce v gridu zapsat metody *setSortable()* a *setFilter()*. Tímto jednoduchým zápisem se pro sloupeček aktivuje seřazování, které se provede po kliknutí na název sloupce, a zobrazí se vyhledávací okno v hlavičce tabulky. Pokud je použito více filtrů, je možné vyhledávat složené výrazy.

Funkce pro vytvoření tabulky s přístroji:

```
protected function createComponentGrid($name)
{
   $nazvySloupcu = $this->pristroje->getColumnName('Pristroje');

   $grid = new Grido\Grid($this, $name);
   $grid->setModel($this->context->pristroje->findAll());

   foreach ($nazvySloupcu as $nazevSloupce)
   {

      if ($nazevSloupce == 'id_pristroj')
      {
         $grid->addColumn($nazevSloupce, $nazevSloupce, Co-
                lumn::TYPE_TEXT)
             ->setSortable();
         $grid->setPrimaryKey($nazevSloupce);
      }
      elseif ($nazevSloupce == 'datum_revize')
      {
         $grid->addColumn($nazevSloupce, $nazevSloupce, Co-
                lumn::TYPE_DATE)
             ->setDateFormat (Grido\Components\Columns\Date::FORMAT_SQL)
             ->setSortable()
             ->setFilter($type = Filter::TYPE_DATE);
      }
      else
      {
         $grid->addColumn($nazevSloupce, $nazevSloupce, Co-
                lumn::TYPE_TEXT)
             ->setSortable()
             ->setFilter();
      }
   }
   $grid->addAction('edit', 'Upravit');
   $grid->addAction('delete', 'Smazat');
}
```

id_pristroj	nazev_pristroje	datum_revize	evidencni_cislo	vyrobni_cislo	lokalizace	Akce
						Vyhledat Zrušit
1	Defibrilátor	2013-04-13	11846/485	51654/46542	xxx	Upravit Smazat
2	RTG	2013-01-20	15118/554	1350	xxx	Upravit Smazat
3	UZV					Upravit Smazat

Položky 1 - 3 z 3 20 Položek na stránku

Obr. 4.3: Tabulka přístrojů.

Na Obr. 4.3 lze vidět výsledný vykreslený datagrid kódem, který je uveden výše. Uživateli je zobrazena kompletní tabulka přístrojů, kterou lze seřadit podle všech parametrů a lze nad ní provádět složené vyhledávací dotazy. Ve sloupečku pro akce má uživatel možnost potvrdit nebo zrušit vyhledávací dotazy a také jednotlivé záznamy upravovat, popř. mazat. Ve spodní části tabulky pak lze zvolit, kolik položek se zobrazí na jednu stranu, popř. procházet více stran, pokud databáze obsahuje více záznamů.

4.4.1 Funkce getColumnName($tableName)

Výše zmíněná proměnná *nazevSloupce* je součástí kolekce s názvem *nazvySloupcu*. Do této kolekce je uložena návratová hodnota funkce *getColumnName($tableName)*. Tato funkce má za úkol zjistit názvy jednotlivých sloupců v dané tabulce, jejíž název se funkci předává jako vstupní parametr. Součásti funkce *getColumnName($tableName)* jsou následující:

- definování připojení k databázi
- připojení k databázi a získání dané tabulky
- cyklus, pomocí kterého se do kolekce zapisují názvy sloupců tabulky

Zdrojový kód funkce vypadá takto:

```
public function getColumnName($tableName)
      {
            $retVal;

            $host = 'rfidexpert.vsb.cz';
            $port = '3306';
            $server = $host . ':' . $port;
            $user = 'vop0003';
            $password = 'rfofthings';

            $link = mysql_connect ($server, $user, $password);
            mysql_select_db('test', $link);
            $i = 0;
            $res = mysql_query('select * from ' . $tableName);

            while ($i < mysql_num_fields($res))
            {
                  $meta = mysql_fetch_field($res, $i);
                  $retVal[$i] = $meta->name;
                  $i = $i + 1;
            }
            return $retVal;
      }
```

4.5 Formuláře

Všechny formuláře jsou vytvářeny pomocí souboru *Form.php* z knihovny Nette. V každém presenteru, který pracuje s formuláři, jsou zapsány alespoň dvě funkce. Jedna vytváří formulář a ve druhé jsou zapsány akce, které se provedou po jeho odeslání. Možnou třetí funkcí je sekvence příkazů při zrušení formuláře. Zde řešeno přesměrováním na výchozí šablonu.

Vytvoření ukázkového formuláře:

```
$form = new Form;
$form    ->addText('jmeno', 'Jméno:')
         ->setRequired('Prosím, zadejte jméno pacienta.');
…
```

První řádek reprezentuje vytvoření formuláře. Dále se zapisují prvky, které má formulář obsahovat. Jedná se o textové pole s popisem (textové pole *jmeno* s popisem *Jméno:*), textové pole pro heslo, checkbox, radiobutton, výběr ze seznamu, prostor pro nahrání souboru a tlačítka s definovanou funkcí. K jednotlivým prvkům lze přiřadit různá validační pravidla. Pokud je

použit parametr *setRequired()*, zadané pole musí být vyplněno, jinak se uživateli zobrazí zpráva a formulář se neodešle. Mezi další pravidla patří omezení délky zadaného textu (minimální, maximální, pevně daná), kontrola platnosti emailu, ošetření zadávání celých čísel, skryté pole atd. Kromě základních validačních pravidel může uživatel definovat svá vlastní.

Zápis tlačítka ve formuláři:

```
$form    ->addSubmit('cancel', 'Zrušit')
         ->setValidationScope(NULL)
         ->onClick[] = $this->formCancelled;
```

První řádek zápisu vytvoří tlačítko *Zrušit*, funkce *set ValidationScope(NULL)* zakáže validaci formuláře (v případě zrušení by se jednalo o zbytečnou akci) a parametr *onClick[]* určí, co se stane, pokud uživatel na tlačítko kline. V tomto případě se zavolá funkce *formCancelled*, formulář se zruší a uživatel je přesměrován na výchozí obrazovku.

Formuláře jsou vytvářeny genericky, všechny sloupce z databáze se do formuláře zobrazí jako textová pole. Ukázkovým příkladem je formulář pro vytvoření nového pacienta, v němž se zároveň používá nejvíce validačních pravidel.

```
protected function createComponentPacientiForm()
{
   $nazvySloupcu = $this->pacienti->getColumnName("Pacienti");
   $form = new Form;

   foreach ($nazvySloupcu as $nazevSloupce)
   {

      if ($nazevSloupce != 'id_pacient')
      {
         if ($nazevSloupce == 'rc')
         {
            $form->addText($nazevSloupce,$nazevSloupce)
                 ->setRequired('Prosím, zadejte' + $nazevSloupce +
                  'pacienta.')
                 ->addRule($form::INTEGER, 'Zadejte rodné číslo jako celé
                  číslo.')
                 ->addRule($form::LENGTH, 'Špatná délka rodného čísla.',
                  array(9, 10))
                 ->addRule(function (Nette\Forms\Controls\TextInput
                  $control) {
                   return (bool) ($this->verifyRC($control->getValue()));
                  }, 'Špatně zadané rodné číslo.');

         }
         elseif (($nazevSloupce == 'kod_pojistovny')||($nazevSloupce ==
            'telefon'))
         {
            $form->addText($nazevSloupce,$nazevSloupce)
                 ->setRequired('Prosím, zadejte' + $nazevSloupce +
                  'pacienta.')
                 ->addRule($form::INTEGER, 'Zadejte' + $nazevSloupce +
                  'jako celé číslo.');
         }
         else
         {
            $form->addText($nazevSloupce,$nazevSloupce)
```

```
                ->setRequired('Prosím, zadejte' + $nazevSloupce +
                 'pacienta.');
        }
      }
   }

   $form->addSubmit('save', 'Uložit')
        ->setAttribute('class', 'default')
        ->onClick[] = $this->pacientiFormSubmitted;

   $form->addSubmit('cancel', 'Zrušit')
        ->setValidationScope(NULL)
        ->onClick[] = $this->formCancelled;

   $form->addProtection();
   return $form;
}
```

V tomto formuláři se na základě názvů sloupců vygenerují všechna textová pole, kromě id pacienta, které nelze zobrazit. Pokud má textové pole název *rc* a jedná se tak o rodné číslo, platí pro toto pole následující pravidla:

- pole musí být vyplněno
- vstupní formát musí být celočíselný (integer)
- délka rodného čísla může být 9 až 10 číslic
- rodné číslo musí splnit validační pravidlo pro rodná čísla

Platnost rodného čísla se ověřuje pomocí samostatné funkce, v níž je zapsán kód pro ověření. Tato funkce byla použita z článku Davida Grudla na webu phpFashion. David Grudl zde ukazuje, že algoritmus pro ověření rodného čísla je jiný než dělitelnost 11: „Spočti zbytek po dělení prvních devíti číslic a čísla 11. Je-li zbytek 10, musí být poslední číslice 0. Jinak poslední číslice musí být rovna zbytku.“ [8]

Další textová pole jsou ošetřena na povinnost vyplnění a na formát vstupní hodnoty (kód pojišťovny, telefon). Jedná-li se o formulář, kde se jednotlivé parametry upravují, je znemožněno uživateli přepsat důležité prvky (rodné číslo, datum narození aj.) přidáním parametru *setDisabled()*, díky němuž je textové pole přístupné pouze pro čtení. Formulář vytvořený předchozím kódem vypadá následovně:

Obr. 4.4: Formulář pro přidání nového pacienta.

4.6 Práce se záznamy

Se záznamy lze provádět tři základní operace – přidávat, upravovat a mazat. Tyto akce jsou dostupné pomocí odkazů v datagridu. Na obrazovce s úvodní tabulkou nalezneme odkaz pro přidání nad touto tabulkou, odkazy pro úpravu a mazání se nachází v řádku se záznamem. Při mazání nebo upravování pracujeme s určitým záznamem, odkazu je přiřazeno id daného záznamu. Dále je možné záznamy seřazovat a vyhledávat. Tyto dvě funkce jsou realizovány přes knihovnu Grido (viz kapitola Tabulky).

Zápis akcí pro úpravu a mazání v datagidu:

```
$grid->addAction('edit', 'Upravit');
$grid->addAction('delete', 'Smazat');
```

Po kliknutí na odkaz se zavolá funkce přiřazená k dané operaci. Po kliknutí na *Upravit* se zavolá metoda *renderEdit*, která určuje další operace. Otevře se daná šablona, případně další formuláře. V případě úpravy záznamu se zobrazí formulář, v němž jsou vyplněny známé údaje. Po odeslání formuláře se záznam v databázi aktualizuje. Při mazání záznamu se zobrazí formulář pro potvrzení smazání.

4.6.1 Vytvoření nového uživatele

Aby do aplikace mělo přístup více uživatelů, lze v databázi vytvořit další uživatelské účty. Po kliknutí na odkaz na spodní straně stránky („Vytvořit nového uživatele zde.“) se zobrazí formulář pro vytvoření nového uživatele (viz Obr. 4.5).

Obr. 4.5: Formulář pro vytvoření nového uživatele.

V zápisu textového pole pro kontrolu hesla se nachází pravidlo, které kontroluje shodu obou zapsaných hesel. Pokud se hesla neshodují, formulář se neodešle. Po zadání všech parametrů a odeslání formuláře se zavolá metoda, která vyplněná data dále zpracovává. V prvé řadě je třeba ověřit, zda se v databázi nenachází uživatel se stejným uživatelským jménem. Toto ošetřuje následující část kódu.

```
if ($key == 'username')
{
      foreach ($users as $user)
      {
            $value = Strings::lower($value);
            $user->username = Strings::lower($user->username);
            if ($user->username == $value)
            {
                  $this->flashMessage('Uživatel již v databázi
                  existuje.');
                  $this->redirect('Dashboard:');
            }
      }
}
```

Pokud takový uživatel neexistuje, do databáze se uloží uživatelské a skutečné jméno a heslo, které se před zápisem zašifruje pomocí hashování.

4.7 Karta pacienta

Tabulka pacientů disponuje ještě jednou funkcí, a tou je zobrazení karty pacienta, neboli zobrazení všech dostupných záznamů o pacientovi (provedená vyšetření, podané léky). Pro tuto funkci je využita akce datagridu s nastavením:

```
$grid->addAction('show', 'Zobrazit kartu pacienta');
```

která uživatele přesměruje na pacientovu kartu. Zde jsou pomocí dvou datagridových tabulek vykresleny záznamy o vyšetření a podání léků pro konkrétního pacienta.

Zápis šablony *show.latte*:

```
{block #content}
<h1 n:block="title">Karta pacienta {$pacienti->jmeno} {$pacienti-
   >prijmeni}, rodné číslo: {$pacienti->rc}</h1>
<h2> Seznam vyšetření</h2>
{control vysetreniGrid}
<br>
<h2> Seznam podaných léků</h2>
{control podaniGrid}
<a n:href="default">Zpět</a>
```

Tato šablona si nechá vykreslit oba datagridy (jeden pro vyšetření a jeden pro podání léků). Uživateli se zobrazí dvě tabulky, které jsou analogicky podobné všem tabulkám v této aplikaci. Jediným rozdílem je fakt, že zde záznamy nelze přidávat (měly by být automaticky načítány pomocí RFID) a při jejich editaci lze upravovat pouze některé parametry.

4.8 Grafika

Tato webová aplikace používá pouze jednoduchou grafiku, jakou je barevné zvýraznění nadpisů, ohraničení a pozadí polí. Důvodem je to, že aplikace musí být pro uživatele především přehledná.

Veškeré grafické úpravy jsou realizovány přes kaskádové styly v souboru *site.css*, který se nachází v adresáři *www*. V tomto souboru jsou definovány grafické styly pro datagridy, formuláře a ostatní prvky. Grafické formátování formulářového prvku ukazuje následující kód, který obsahuje formátování i pro různé části formulářů (popisky textových polí, textové pole, aj.).

Ukázka ze souboru *site.css* – formátování formuláře:

```
form {
      max-width: 300px;
      padding: .8em 1.6em;
      background: #ff9999;
      border: solid 2px #ff6666;
}

form input {
      margin: 2px 0;
      font-size: 100%;
}

form input.default {
      font-weight: bold;
      font-size: 105%;
}

form input.text {
      padding: 4px 2px;
      border: solid 1px #add4fb;
      max-width: 105px;
      max-height: 15px;
}

form label {
```

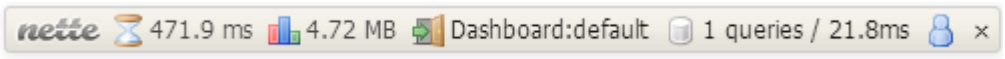
```
        width: 150px;
        display: block;
        text-align: right;
        margin-right: 5px;
        font-weight: normal;
}

form .required label {
        font-weight: bold;
}
```

4.9 Testování a optimalizace

Při spuštění aplikace se ve spodním pravém rohu nachází tzv. debug bar neboli ladicí lišta (viz Obr. 4.6). Tato lišta poskytuje informace o stavu a časové náročnosti aplikace. Po najetí kurzoru na určitou část lišty se zobrazí konkrétní parametry, mezi něž patří:

- informace o verzi PHP, Apache serveru, Nette Frameworku
- čas potřebný k načtení stránky
- maximální velikost přidělené paměti
- název aktuální zobrazené šablony
- počet SQL dotazů a doba jejich trvání
- informace o přihlášeném uživateli



Obr. 4.6: Ladicí lišta (debug bar).

Díky této liště lze jednoduše zjistit rychlost načítání aplikace i jednotlivých stánek webové prezentace a následně tuto aplikaci optimalizovat. Optimalizovat aplikaci lze snížením její grafické náročnosti, zjednodušením použitých SQL dotazů a také použitím různých ovladačů k přístupu k databázi. Tato aplikace byla testována na dva použité ovladače: ovladač *mysql* a ovladač *mysqli*. Testování proběhlo nad tabulkou *Pacienti*, která obsahuje cca 20 000 záznamů, při vyhledávání v tabulce (filtrování záznamů).

Queries: 2, time: 62.519 ms

Time ms	SQL Statement
28.692 explain ▶	SELECT COUNT(*) FROM `Pacienti` WHERE ((`rc` LIKE ?)) ...\libs\Grido\DataSources\NetteDatabase.php
33.827 explain ▶	SELECT `id_pacient`, `rc`, `jmeno`, `prijmeni`, `datum_narozeni`, `kod_pojistovny`, `pohlavi`, `mesto`, `obec`, `ulice`, `cp`, `telefon` FROM `Pacienti` WHERE ((`rc` LIKE ?)) LIMIT 100 OFFSET 0 ...\temp\cache_Nette.FileTemplate_Grido.Grid.latte-135a820abd52e8db2dd963ef6038f001.php

Obr. 4.7: Výsledek vyhledávání ve sloupci rodné číslo.

Obr. 4.7 zobrazuje počet SQL dotazů a dobu jejich trvání při filtraci rodného čísla. V datagridu je zobrazeno 100 záznamů a rodné číslo mělo obsahovat hodnotu 77. Lze vidět, že databáze byla schopná vrátit výsledek za 62.519 ms, přičemž samotný vyhledávací dotaz

na rodné číslo trval 33,827 ms. Těchto hodnot bylo dosaženo při použití ovladače mysql. Ovladač mysqli byl schopný provést dané dotazy přibližně od 10 ms rychleji, proto byl po testování pro aplikaci zvolen tento. Co se týká komplexních vyhledávacích dotazů, jejich provedení probíhá delší dobu. Přidá-li se kromě vyhledávání podle rodného čísla také vyhledávání čísla telefonního, celkový dotaz je vykonán za cca 71 ms, což je zhruba o 10 ms déle než vyhledávání rodného čísla. V reálném čase jsou však tyto rozdíly uživatelem stěží postřehnutelné.

Rychlost databázového vyhledávání je také zajištěna použitím datagridů, které umožňují zobrazit záznamy v menším počtu na více stránek. Pro nastavení dvaceti záznamů na stránku se v databázi najde pouze prvních dvacet záznamů, a to platí i pro použití vyhledávání. Zobrazování tak neprochází celou databází a je tak zefektivněna práce celé aplikace.

5 Závěr

Hlavním cílem této bakalářské práce bylo navrhnout a vytvořit aplikaci schopnou spravovat data z oddělení urgentního příjmu v Ostravě-Porubě. Prvním úkolem bylo seznámit se s prostředky pro tvorbu webových aplikací, naučit se používat jazyk PHP, databázový systém MySQL, zjistit funkci serveru Apache a získat poznatky o využitelnosti frameworku Nette. Základem praktické části byla instalace potřebných softwarových produktů (program EasyPHP pro spouštění serverů Apache a MySQL, program Eclipse jakožto vývojové prostředí aplikace).

Následně byl analyzován návrh aplikace a s přihlédnutím k požadovaným funkcím a k použití pro nemocniční prostředí byla vytvořena testovací databáze pro pacienty a ostatní základní entity. Z důvodu nedostatečné komunikace s vedením urgentního příjmu Fakultní nemocnice Ostrava se tato databáze omezila pouze na několik málo tabulek. Po úspěšném dokončení části projektu, v níž se navrhuje celá databáze, může být tato aplikace aktualizována a nasazena na tuto kompletní databázi, jak bylo původním plánem. Princip funkce aplikace a použité prvky jsou však stejné i pro použití na testovací databázi. Pro funkci aplikace byl vytvořen diagram aktivit.

Samotná realizace aplikace se prováděla pomocí frameworku Nette, který také díky své bohaté uživatelské dokumentaci značně zefektivnil vývoj aplikace. Co se týká bezpečnosti, aplikace je chráněna uživatelským přihlašováním, kde v databázi jsou uložena hesla zašifrovaná pomocí hashovací funkce. Hlavní stránkou aplikace je tabulka pacientů. Je tomu tak proto, že pacienti jsou z hlediska nemocniční péče nejdůležitější entitou. Aplikace umožňuje všechny požadované práce se záznamy (přidávání, upravování, mazání, hledání záznamů) na všech tabulkách v testovací databázi.

Aplikace vytvořená v rámci této bakalářské práce je součástí projektu pro urgentní příjem Fakultní nemocnice Ostrava. Celý projekt by měl přispět k zvýšenému monitorování pacientů a ke zjednodušení vedení nemocniční dokumentace. V rámci jiných bakalářských prací byla navržena kompletní databáze a také Java aplikace pro mobilní zařízení, která s touto databází komunikuje.

Prostor pro vylepšení aplikace se nachází zejména v použití AJAXu. Aplikace by se pak stala více interaktivní a byla by více dynamická, což by přispělo k lepší uživatelské práci. Příkladem může být jednořádková editace záznamů bez přesměrování na jinou stránku. Dalším zdokonalením aplikace by pak bylo její rozšíření na celou nemocniční databázi s použitím více tabulek a více záznamů. Při tomto nasazení by také bylo možné aplikaci lépe testovat a optimalizovat.

Z mého osobního hlediska by aplikace po dalším vývoji mohla být reálně využitelná pro nemocniční praxi. A protože je tato aplikace univerzální a generická, může najít uplatnění i kdekoliv jinde.

Seznam literatury

[1] **Darie, Christian, a další.** *AJAX a PHP – tvoříme interaktivní webové aplikace profesionálně.* [překl.] Roman Skřivánek. Brno : Zoner Press, 2006. ISBN 80-86815-47-1.

[2] **Boronczyk, Timothy, a další.** *PHP 6, MySQL, Apache: Vyvářime webové aplikace.* [překl.] Bogdan Kiszka. Brno : Computer Press, 2009. ISBN 978-80-251-2767-4.

[3] PHP: History of PHP - Manual. *PHP: Hypertext Preprocessor.* [Online] 2013. [Citace: 14. leden 2013.] http://php.net/manual/en/history.php.php.

[4] January 2013 Web Server Survey | Netcraft. *Netcraft.* [Online] Netcraft Ltd., leden 2013. [Citace: 9. leden 2013.] http://news.netcraft.com/archives/2013/01/07/january-2013-web-server-survey-2.html.

[5] About the Apache HTTP Server Project - The Apache HTTP Server Project. *The Apache HTTP Server Project.* [Online] The Apache Software Foundation, 2012. [Citace: 9. leden 2013.] http://httpd.apache.org/ABOUT_APACHE.html.

[6] **Pošmura, Vlastimil.** *Apache – Příručka správce WWW serveru.* Praha : Computer Press, 2002. ISBN 80-7226-696-9.

[7] MVC aplikace & presentery | Nette Framework. *Nette Framework.* [Online] Nette Foundation, 2013. [Citace: 15. leden 2013.] http://doc.nette.org/cs/presenters.

[8] **Grudl, David.** Jak ověřit platné IČ a rodné číslo? *phpFashin.* [Online] 2007. [Citace: 30. duben 2013.] http://phpfashion.com/jak-overit-platne-ic-a-rodne-cislo.

Seznam příloh

Přiložené CD obsahuje:

- vypracovanou bakalářskou práci ve formátu .pdf
- zdrojové kódy vytvořené aplikace